AMENDMENTS TO THE SPECIFICATION

Please amend paragraph [0015] on page 6 as follows:

[0015] It is desired in some situations to run the subject code 10 on the target machine 2 of the present invention, which includes a target processor 22 using a set of target registers 24. The two processors 12 and 22 of the subject machine 1 and the target machine 2, respectively, may be inherently non-compatible, such that these two processors 12 and ~~24~~22 use different instruction sets. The target processor 22 includes a floating point unit 28 for computing floating point operations and an integer unit 26 for performing integer operations. The floating point unit 28 and the integer unit 26 may comprise any of a wide variety of types of hardware units, as known to those skilled in the art, where the floating point unit 28 is preferably IEEE 754 Standard compatible floating point hardware.

Please amend paragraph [0019] on page 8 as follows:

[0019] where a, b, c and d are operands which can be expressed as floating point numbers. High precision as referred to in this description means any precision which is higher than that provided by the target machine 2. For instance, if the architecture [[p]]of the target machine 2 supports IEEE Standard 754 double-precision floating point values, then high precision would refer to any values having a higher precision than double-precision floating point values. It should be noted that the floating point emulator 20 only calculates the intermediate values of the accumulated instructions at high precision, and the operands themselves and the result are not at high precision.

Please amend paragraph [0024] starting on page 9 as follows:

[0024] If none of the operands (a, b, c) are special values, it is next determined in step 206 whether the exponent for the result of the multiplication (a*b) overlaps with the exponent of operand c. Two values will overlap if the addition/subtraction of the significant digits of the two values yields a result different from each of the two values. In this context, non-overlapping refers to the fact that either a*b or c is so large as to make the other insignificant. By way of example, in the situation where a particular FPU is only capable of representing 3 significant digits. If the value 3.10 is added to the value to 0.01, then it can be seen that both values are important to the result, i.e., performing the addition will yield a result different to the sources and the sources thus overlap. Contrarily, for the same FPU only capable of representing 3 significant digits, if the value 310 is added to the value 0.01, the result when using 3 significant figures is 310. Thus, in this situation the result is the same as the first source and the two values did not overlap.

Please amend paragraph [0041] starting on page 9 as follows:

[0041] It is known that a double-precision floating point value has 52 bits for its mantissa (plus the implied 1), thus man(x) ~~it~~ is 53 bits wide. The result of the multiplication will therefore be a maximum of 106 bits wide. It is then determined if the extra precision is required by examining the spread of the resulting mantissa. If this mantissa would fit within a float double (i.e., 53 bits including the implied one), then the extra precision is not required. This is tested by checking to see if the bottom 53 bits of the resulting mantissa were used.

```
---------------------------------------------------------------------------
        If ((man(a*b) & 0x1fffffffffffff) = = 0)
                FPU(a*b – c)
    EndIf

            exp(a*b) = exp(a) + exp(b)
          sign(a*b) = sign(a) xor sign(b)
---------------------------------------------------------------------------
```

It is now necessary to align a*b and c, in order to perform the subtraction.

```
---------------------------------------------------------------------------
    If (exp(a*b) > exp(c))
        Shift_right(man(c), exp(a*b) - exp(c))
        exp(a*b-c) = exp(a*b)
    Else
        Shift_right(man(a*b), exp(c) - exp(a*b))
        exp(a*b-c) = exp(c)
    EndIf

    If (man(a*b) > man(c))
        sub(man(a* b), man(c))
        sign(a*b-c) = sign(a*b)
    Else
        sub(man(c), man(a*b))
        sign(a*b-c) = sign(c)
    EndIf
---------------------------------------------------------------------------
```

The resulting mantissa is then checked to see if it equals zero.

```
---------------------------------------------------------------------------
    If (man(a*b-c) = = 0)
        FPU(0.0)
    EndIf
---------------------------------------------------------------------------
```